A FP-CNN method for aircraft fault prognostics

Zhiyu Chen[1, a)] Lihong Shang[1, b)], and Mi Zhou[2, c)]

[1]*School of computer, Beihang University, beijing 100191, China.*
[2]*Reliable Avionics Technology Co. Ltd., Beijing 100191, China.*

[a)] carloschen@buaa.edu.cn
[b)] shanglh@buaa.edu.cn
[c)] zoneme@sina.com

Abstract. Predicting the status of flight vehicle in advance can have huge advantages in maintenance and early warning areas. Accurate forecast helps reduce maintenance costs and improve safety during the aircraft's life cycle. Combining the ability of convolutional neural network to extract features of different levels and its computational efficiency, a novel convolutional neural network -- fault prognosis convolutional neural network(FP-CNN) is proposed in this paper, the purpose of which is to predict the Remaining Useful Life (RUL) by learning sequential information and extracting sensor features from noisy datasets under different operating modes. An experiment on CMPASS data is conducted to prove the efficiency and accuracy of this framework.

INTRODUCTION

With the development of flight vehicle, an increasing number of on-board sensors are planted in the vehicle. Accessing and storing sensor data becomes handier due to sensor techniques. The large amount of sensor data containing information about the status of working vehicles has made data-driven condition monitoring system more popular than before. Traditionally, people used physical models like wavelet transform, empirical mode decomposition and short-time Fourier transform. But in current practice, those methods struggle in dealing with mass data [1, 2]. As artificial intelligence as an approach has achieved great success in many fields, machine learning algorithms started to be applied in fault diagnosis and have had promising results [3].

It is a great challenge to diagnose failure using sensor data. First, one flight vehicle could have thousands of sensors with all sensor data sampled and stored at a particular rate during the operation of the equipment. Although those data contain information about system conditions, their dimension is high and health information is not easy to extract. Second, not all sensor data is contributed to the system's health condition. In fact, only a few sensors play a key role in system failure [4]. All the other data of the device can be considered as noise. Third, raw sensor values cannot be directly compared with each other because the vehicle system has many operation phases. One value may indicate different system statuses in different operation models [5].

Expert knowledge used to be the major basis of the pre-determination of useful and robust sensor values. For example, Nawaz et al [6] applied Bayesian-network-based inference system in the etching manufacturing process, which was based on the expert knowledge about the relationship between root causes, equipment and process parameters. Mahadevan et al [7] employed one-class SVM to detect odd behaviors during semiconductor etching process. You et al [8] applied principal component analysis (PCA) to extract features from acquired signals before using them as inputs for feedforward neural network. This model succeeded in detecting defects. Liu et al [9] proposed a time-frequency dictionary which could produce different identities for different raw vibration dat. Those mapped vectors were then used as SVM inputs for the diagnosis of bearing faults. Kang et al [10] employed relative energy in a wavelet packet node (REWPN) and entropy in wavelet packet node (EWPN) as support vector machines (SVMs) for diagnosis and fault classification.

Though traditional machine learning algorithms could deal with mass data and reduce the negative effects of noise and high dimensions, they needed professional knowledge about specific fields and could not extract complex non-linear feature information from massive data. Deep learning, however, is able to fill those gaps, thus making a big splash in recent years. Compared with traditional machine learning algorithms, deep-learning-based networks have more layers to extract features automatically. After the extraction by several layers, the raw input values are turned into more abstract features without any irrelative information. Deep learning has proved to be a huge success in Computer Vision [11] and speech recognition [12]. Wang et al [3] adopted deep belief network (DBN) on health state classification, which is more accurate than SVM or SOM. Li and Zhang [13] et al proposed a novel fully-connected winner-take-all autoencoder network to detect bearing faults. They imposed lifetime sparsity on the encoded features by keeping a maximum of k% neurons activated by each layer. This method turns out to be more accurate than the conventional deep neural network (DNN). Ince et al [14] proposed 1-D convolutional neural network to detect motor faults. Zhang et al [15] employed training interference-convolutional neural network which could reduce the noise of raw data and detect bearing faults under different operation models. Pan et al [5] used a novel deep learning network (LiftingNet) to learn features adaptively from raw mechanical data without prior knowledge. Results showed that this method could achieve layer-wise feature learning and classify mechanical data with random noise. Lee et al [16] proposed a convolutional fault detection and classification neural network (FDC-CNN), in which a receptive field was tailored to multivariate sensor signals slides along the time axis to extract fault features.

Despite great improvement in fault diagnosis, those methods detect faults only when they have occurred. For higher efficiency on aircraft maintenance, more advanced techniques are needed to predict when faults will take place. Fault prognostics is a more promising research area, enabling people to make decisions before failure occurs by predicting the health condition of the system. But due to the complexity of system failure degradation, there are few researches on fault prognostics.

This paper proposes a convolutional neural network called FP-CNN to estimate system's remaining useful life. Section 2 introduces the background information about convolutional network. Section 3 presents the main idea of the proposed framework. And Section 4 first introduces the experiment data, then fixes the network with detail parameters and functions. The results and analyses are presented in the last part of Section 4.

BACKGROUND

Standard convolutional neural network

Convolution neural network (CNN) has become the most popular algorithm in deep learning field due to its powerful learning ability in image classification, text recognition and so on. It is a neural network designed to handle data with a grid type of structure. Timing data, for example, can be considered as a one-dimensional grid on the time axis formed by samples. Image data is typical two-dimensional grid data. Convolution is a special type of linear operation. Convolutional networks are neural networks that use convolution instead of normal matrix multiplication at least in one layer of the network.

As shown in Fig. 1, a standard convolution neural network consists of five parts: input layer, convolutional layer, subsample layer, fully-connected layer and output layer. Convolutional layer is the core of the whole network where several filters are applied to the input data. An input value is used for many different types of feature extraction. The number of filters in the layer is equal to the amount of features extracted from the input data. Another important part is the subsample layer (pooling layer). This layer is responsible for reducing the size of the data. The last pooling layer or convolutional layer is usually connected to one or more fully connected layers, the output of which is the final output of the model.

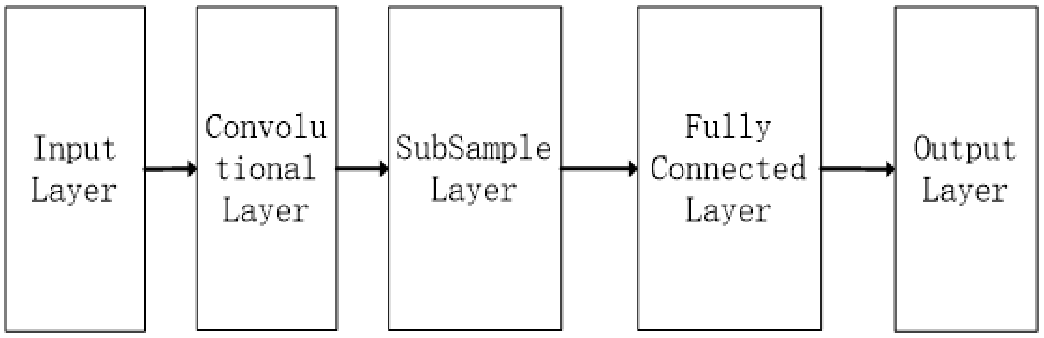

FIGURE 1. A standard convolutional neural network

Convolutional layer

The input layer receives data or mostly images, then the operation involving the kernel function and the input data is carried out to extract local features and store those features as a feature map. As shown in Fig. 2, a kernel function is a matrix (normally square) of weights with a much smaller size than the input data. The feature map has many nodes, each representing a certain input data area operating on the kernel function. The kernel functions operate across the entire input data with specific steps in both horizontal and vertical directions. The following formula (1) is a two-dimension convolutional operation:

$$y_{ij} = \sigma\left(\sum_{m=1}^{L_m} \sum_{n=1}^{L_n} w_{mn} x_{(r+i\cdot s)(c+j\cdot s)} + b\right) \tag{1}$$

$$0 \le i \le \frac{H - L_m}{S} + 1, 0 \le j \le \frac{W - L_n}{S} + 1$$

y_{ij} is the value of the kernel function operating on a particular area. σ is an activation function. L_m and L_n are the height and width of the kernel matrix respectively. w_{mn} and $x_{(r+i\cdot s)(c+j\cdot s)}$ represent the coordinates of the kernel matrix and the input data respectively. b is the bias of the equation. H and W are the height and width of the input data respectively. S denotes the length of the kernel step; and i and j stand for the feature map's height and width respectively. The size of the output feature map is $((H-L_m)/S +1) * ((W-L_n)/S+1)$.

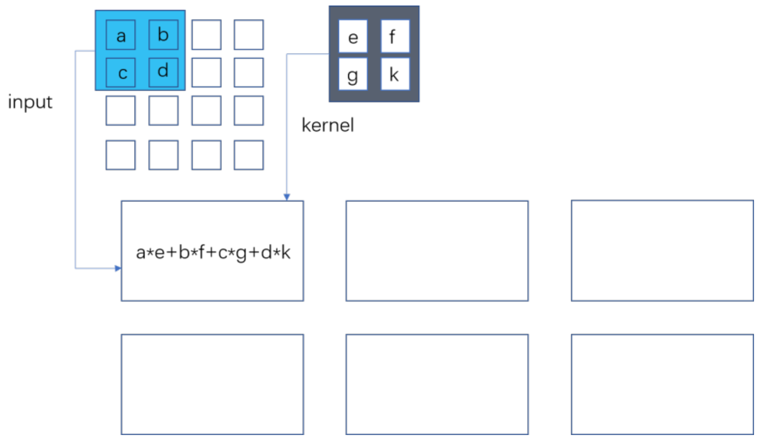

FIGURE 2. An example of two-dimensional convolution. Boxes with arrows are used to illustrate how the upper-left corner of the output tensor is convolved by applying the kernel to the upper-left corner of the input tensor

Convolution operation helps improve machine learning through three main concepts: sparse interaction, parameter sharing and equivariant representation [17]. In traditional neural networks, each input neural node interacts with every output neural node. The size of the parameter matrix grows with the amount of these nodes, requiring more sophisticated calculation. The sparse interaction of the convolutional network can make the kernel matrix much smaller than the input size. For example, the input image could have millions of pixels, but we can use a kernel of only a few hundred pixels to extract meaningful features. This means we only need fewer parameters and less computation.

Parameter sharing means that we use the same parameters of several functions in one model. In conventional neural networks, the parameters of the weight matrix will be used only once, while in convolution networks, the parameters of the kernel matrix will be used many times. The matrix will operate on every part of the input data. Fig. 3 shows how parameter sharing reduces the computation of the network.

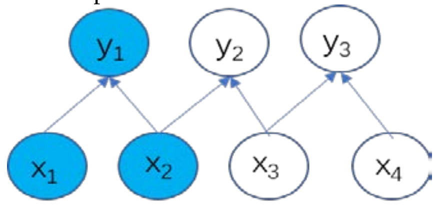

FIGURE 3. Sparsely connected, the blue nodes show how input cells x_1 and x_2, receptive fields of y_1 affect output cells y_1, y is generated by the convolution with a kernel width of 2, and only two input cells affect each cell of output y.

Subsample

Pooling layer usually comes after the activation function to adjust the latter's output. The pooling function uses the local statistical characteristic of the neighboring data as the output of this location. In practice, people normally use max pooling, average pooling and so on. As illustrated in Fig. 4, max pooling takes the maximum value of the adjacent matrix area of a given size as the output of this area.

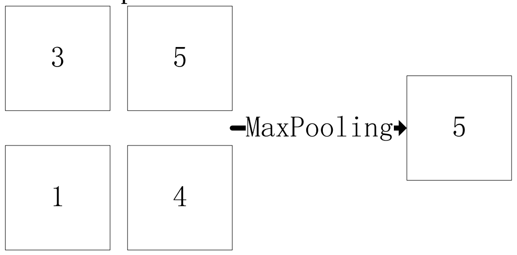

FIGURE 4. 2*2 Max Pooling layer

Pooling is important for handling input data of different sizes. In general, the size of a model's output layer stays unchanged. By adjusting the size of the pooling area, the last layer can get the same dimension of abstract features despite the size of the input.

FP-CNN

Artificial intelligence-based approaches can be roughly divided into supervised and unsupervised learning. The method this paper proposes is based on supervised learning. Supervised learning aims to find a mapping function of input values and labelled targets. The difference between desired target values and mapping function's output is measured and given as feedback to modify the model. Eventually we will get an optimal solution.

Timing data is arranged in a 1-dimensional grid along the time axes in most diagnostics and prediction cases. Full length kernels are used sometimes to extract features of the whole time sequence. Focusing only on the extraction of full-length abstract features of the whole sequence, this approach ignores low-level timing features of several adjacent sequences. In other cases, several layers of small length kernels of different sizes are applied. Those kernel functions can map characteristic information to adjacent time layer-by layer and extract both low-level and high-level features. However, this method requires a lot of extra computing resources.

FP-CNN framework

The structure of our CNN framework is called fault prognosis convolutional neural network(FP-CNN) which is different from that of the traditional CNN model used for diagnostic classification. As depicted in Fig. 5, this model contains an input layer, four convolutional layers and one fully-connected layer. In image classification, channels of input layer are fixed and divided by RGB. In our framework, the number of channels is subject to changes, depending on the dimension of input data. For example, if there are four sensors in one sampling period, the input layer will have four channels. Each channel is a two-dimensional grid. Every pixel of grid represents a value of sensor. $\{T_1^k, T_2^k, T_3^k, \ldots, T_{length}^k\}$ are placed in the first column and $\{T_{1+length*(length-1)}^k, T_{2+length*(length-1)}^k, \ldots, T_{length*length}^k\}$ in the last column of k_{th} channel. Be noticed that the data along the row direction is not sequential in time.

The first layer of the convolution aims to extract incomplete feature information about sensor data in the same cycles. To this end, we employ a kernel of a special size, the length and width of which are both 1. This kernel extracts feature information on the same pixel in different channels, which means we generate basic traits from different sensors from the same cycle. Unlike the traditional convolutional networks, the subsample layer is not added here

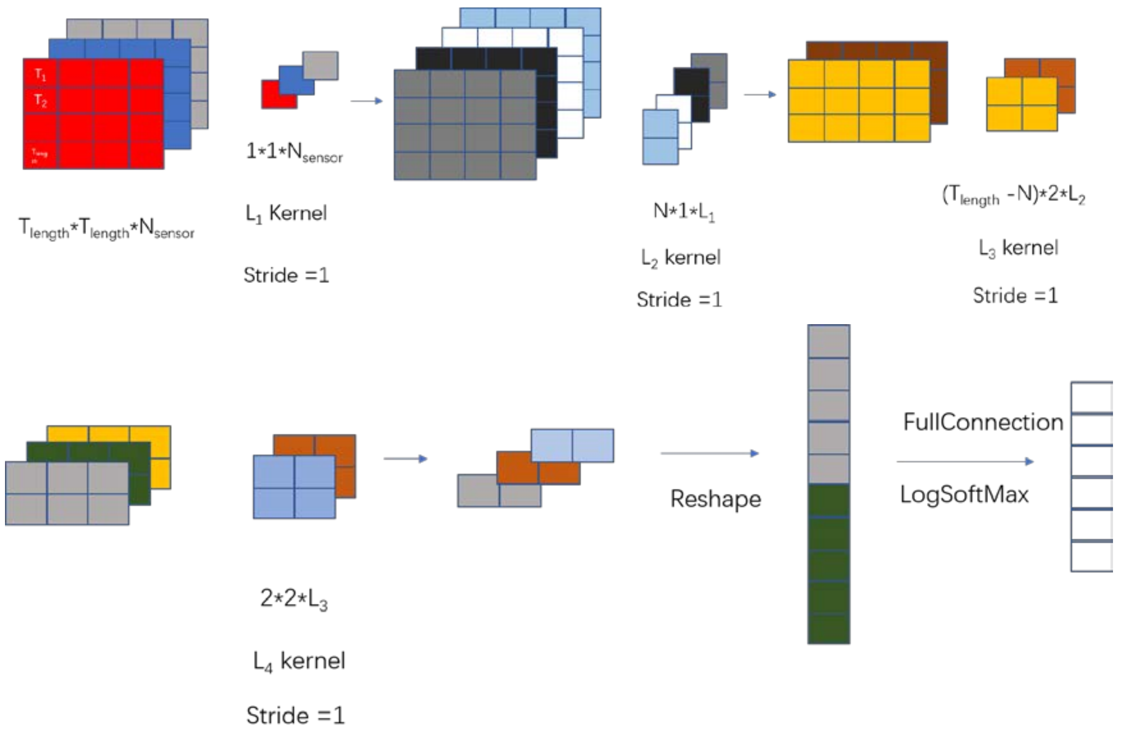

FIGURE 5. Structure of the proposed convolutional neural network

because every pixel of channel grid represents the character of different sample cycles. If pooling layer is applied, we will lose the time accuracy of features.

After convolutional layer, Rectified Linear Units (ReLU) are used as nonlinear activation function defined as formula 2:

$$\sigma(x) = max(0, x) \tag{2}$$

Research [18] showed that the deep CNN with ReLU trained faster than the same framework that used Tanh as their activation function. Also, when we trained our network with gradient descent, ReLU could alleviate the gradient vanishing problem [19].

Now the number of channels equals to the amount of previous convolutional layer's kernel functions. In this convolutional layer we want to generate basic features of neighbor sequences. As mentioned above, the period is continuous in the vertical direction of the channel. In order to avoid the periodic discontinuity in the horizontal direction, we use N*1 filter to extract low-level feature information about N successive sampling cycles.

The next layer is still a convolutional layer which will map abstract high-level timing traits. The size of the filter will be (L-N+1) * M. L denotes the length of the channel grid, and N represents the height of previous layer's kernel function. M and N are hyper-parameters of this framework. Only in this way can the receptive area of the channel be a successive time cycle. Each pixel is coupled with space and time.

The last convolutional layer aims to extract high-level features of input data. We adopted the normal filter of 2*2. We can learn high-level abstract features of time and space within a certain range.

A fully connected layer comes after all the convolutional layers, the equation of the fully-connected layer is shown in formula 3:

$$O = \sigma(Wx_c + b) \tag{3}$$

O represents the output, σ is the Sigmoid activation function, W denotes the parameter matrix of the linear layer, and x_c is the output of convolutional layers. Then LogSoftMax is applied to predict the possibility of each bit. LogSoftMax is depicted in formula 4. At last, the LogSoftMax is used for the output of entire network.

$$P_i = log\frac{e^{O_i}}{\sum_i^N e^{O_i}} \tag{4}$$

CASE STUDY

In this section, the experiment which tests the proposed framework will be presented in three parts. First, the source and characteristics of the experimental data is described. Second, several parameter groups are applied to the framework. Finally, the result of the experiment is discussed.

Input data

The experimental data in this paper comes from the data set released by NASA AMES Laboratory. CMAPSS is a tool for simulating a realistic large commercial turbofan engine. This data set collects periodic sensor information that simulates turbofan engine operation until failure. Turbofan engine is the core component of the aircraft, and is the largest and the most complex subsystem on the aircraft. It is important to know the status of the subsystem in advance.

The data set is composed of 216 units. They are different instances of the same system from different initial states until failure. The unit data is composed of periodic sampled sensor signals. Each cycle consists of unit name, period, 3 operation settings and 21 sensor values. Although it is not known what the physical attenuation model of the turbine engineer system is, it is considered that the degradation has common characteristics. The cycle of each unit is not fixed. The shortest unit has 127 cycles, the longest 356 cycles, and the average number of cycles is 209.

The first step of the experiment is to normalize the data set to have zero mean and standard deviation to ensure sensor data of different ranges is put into the same scope. The equation is depicted in formula 5

$$x_{norm}(n) = \frac{x(n)-\frac{1}{N}\sum_i^N x(i)}{\sqrt{\frac{1}{N}\sum_n^N\left(x(n)-\frac{1}{N}\sum_i^N x(i)\right)^2}} \tag{5}$$

where x(n) is the sampled data of one sensor. Those normalized sensor data is the input of neural network we proposed in this paper.

Network training

Supervised deep learning method was applied to the established CNN model to predict the remaining useful life (RUL) of the system. The model was set to consist of four hidden convolutional layers with several feature maps. There would be fewer units as the model goes deeper due to parameter reduction. The main parameters of the CNN model we proposed are listed in Table 1.

TABLE 1. model parameter size.

Model parameters	**Input layer**	**Convolutional layer 1**	**Convolutional layer2**	**Convolutional layer3**	**Convolutional layer4**
Size of feature map	10*10	1*1	2*1	8*2	2*2

Our input grid consists of 10*10 two-dimensional grids. We set the first convolutional layer has 12, 32, 64, 128 kernels with size of 1*1. The kernel of the second convolutional layer is set to have a size of 2*1, and the number of kernels is 12, 24, 32, 64 and 128 respectively. The third convolutional layer uses 8*2 kernel with the amount of 12, 24, 32, 64 and 128. The last convolutional layer uses 2*2 kernel with the amount of 12, 24, 32, 64 and 128. The output of the fully connected layer is the material for prediction.

We minimize the loss function between the output of the network and target, which is shown in formula 6 where the target is the real value of the labels. The value of the correct bit is 1 and the reset of bits is set to 0

$$L(x_i, target) = -x[target] \tag{6}$$

The framework is trained by root mean square prop (RMSprop). The process of this optimization function is depicted in formula (7) (8) (9) (10):

$$g = \frac{1}{m}\nabla_\theta \sum_i L(x_i, target) \tag{7}$$

$$r = \rho r + (1-\rho) g \Theta g \tag{8}$$

$$\Delta\theta = -\frac{\epsilon}{\delta+\sqrt{r}}\Theta g \tag{9}$$

$$\theta = \theta + \Delta\theta \tag{10}$$

There are several hyper-parameters and learnable parameters: global learning rate ε, initial parameters θ, numerical stability δ, and decay rate ρ. Compared with the traditional Adagrad optimization function, this method is a better solution to the problem of early stop of deep learning.

All network matrix parameters are initialized layer by layer through a zero-mean standard uniform distribution called Xavier initialization. The formula is as follows formula 11:

$$W \sim U[-\frac{\sqrt{6}}{\sqrt{Nin+Nout}},\frac{\sqrt{6}}{\sqrt{Nin+Nout}}] \qquad (11)$$

Result

Among all hyper-parameters to be determined in the training of convolutional neural network, the learning rate represents the rate of updating the parameter weights. In this experiment we used the values ranging between 10e-5 and 10e-2. The exponent was increased by 0.5 each time. As shown in Fig. 4, L1, L2, L3, L4 are the key parameters of the model. Four sets of such values were set in Table 2.

Table 2. Hyper-parameter sets

Experiment sets	L_1	L_2	L_3	L_4	Output length
1	24	10	10	10	150
2	24	24	24	24	200
3	48	48	48	48	250
4	24	64	64	128	300

Fig. 6 shows the testing result of different parameter sets in Table 2. This test unit contains the whole run to fail cycle data.

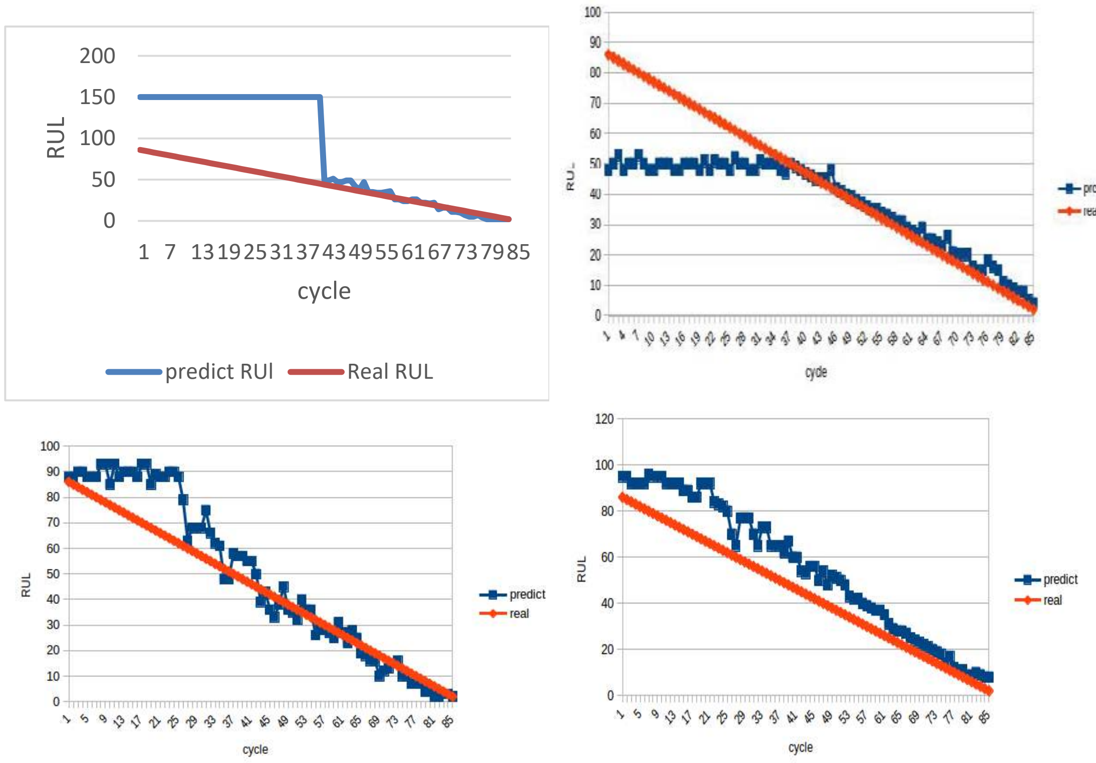

FIGURE 6. picture A B(upper-right) C D for Parameter set 1, 2, 3, 4.

As seen in picture A, at the beginning of the unit instance, there is a big gap between the predicted RUL and real RUL. This is because the predicted RUL 150 does not represent the precise cycle. It means the system does not show any sign of degradation by then. As shown in Table 2, in hyper-parameter set 1 we use 150 as the output length. And as we label the training and test data, if the unit has more than 150 cycles, we label those cycles which have more than 150 cycles to run until failure occurs. Moreover, we should pay more attention to the stages when the fault has already affected the performance of the system, and less attention to the status of the early conditions of the system.

We choose 30 units each of them has run hundreds cycles as test data, the average cycles of those 30 unit is 190. Each unit will fail after less than 50 cycles. The predict result is shown in Fig 7. Each blue column represents the predicting result of a test unit, and the corresponding yellow column is the real RUL of this unit.

The average error between predict RUL and real RUL of this 30 units is 5.5 cycles. This result is better than the result of the first place in the Data Challenge Competition organized by the PHM conference.

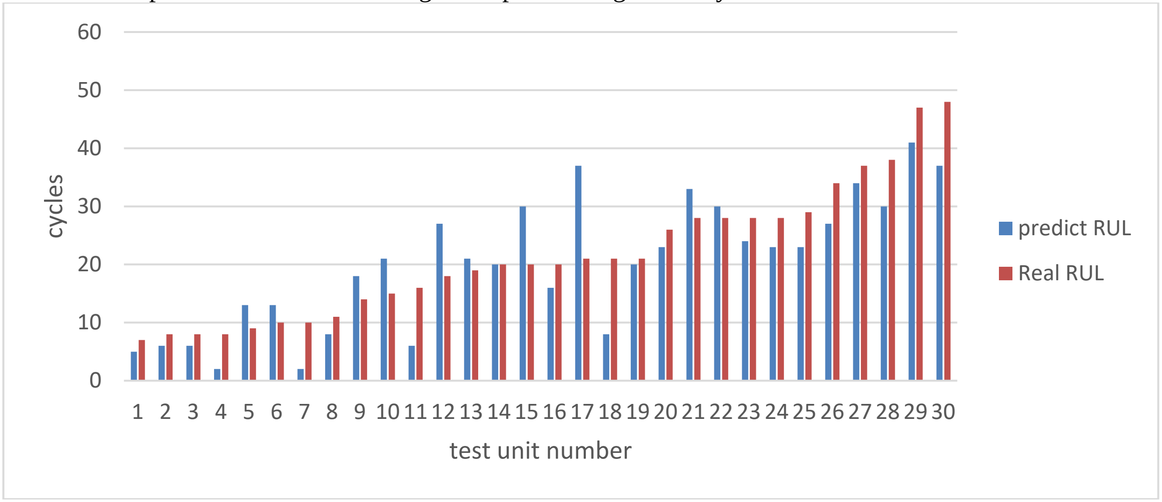

FIGURE 7. The test result of the 30 units

CONCLUSION

In this paper, a new convolutional neural network called FP-CNN is proposed for fault prediction in flight vehicle system. This method starts with raw sensor data; learns sensor dependent features and timing features through convolutional layers; and predicts the remaining useful life of aircraft components. We proved the accuracy of the proposed method by experiments. This method does not need pre-determined feature selection to eliminate the useless sensor values, and is more accurate than the other methods coping with the same issue.

There is still a lot to be done to improve the current framework. We wish to apply deep convolutional neural network to big data on different system components under different fault degradation models in the future so as to train a system-level prognostic system.

REFERENCES

1. Yan R, Gao R X, Chen X. Wavelets for fault diagnosis of rotary machines: A review with applications[J]. Signal Processing, 2014, 96(5):1-15.
2. Lei Y, Lin J, He Z, et al. A review on empirical mode decomposition in fault diagnosis of rotating machinery[J]. Mechanical Systems & Signal Processing, 2013, 35(1–2):108-126.
3. Tamilselvan P, Wang P. Failure diagnosis using deep belief learning based health state classification[J]. Reliability Engineering & System Safety, 2013, 115(7):124-135.
4. Längkvist, Martin, Lars Karlsson, and Amy Loutfi. "A review of unsupervised feature learning and deep learning for time-series modeling." Pattern Recognition Letters 42 (2014): 11-24.
5. Pan, Jun, et al. "LiftingNet: A Novel Deep Learning Network with Layerwise Feature Learning from Noisy Mechanical Data for Fault Classification." IEEE Transactions on Industrial Electronics (2017).
6. Nawaz, Javeria Muhammad, Muhammad Zeeshan Arshad, and Sang Jeen Hong. "Fault diagnosis in semiconductor etch equipment using Bayesian networks." Journal of Semiconductor Technology and Science 14.2 (2014): 252-261.
7. Mahadevan, Sankar, and Sirish L. Shah. "Fault detection and diagnosis in process data using one-class support vector machines." Journal of process control 19.10 (2009): 1627-1639.

8. You, Deyong, Xiangdong Gao, and Seiji Katayama. "WPD-PCA-based laser welding process monitoring and defects diagnosis by using FNN and SVM." IEEE Transactions on Industrial Electronics 62.1 (2015): 628-636.
9. Liu, Ruonan, et al. "Time-frequency atoms-driven support vector machine method for bearings incipient fault diagnosis." Mechanical Systems and Signal Processing 75 (2016): 345-370.
10. Kang, Myeongsu, et al. "Reliable fault diagnosis for low-speed bearings using individually trained support vector machines with kernel discriminative feature analysis." IEEE Transactions on Power Electronics 30.5 (2015): 2786-2797.
11. LeCun, Yann, Yoshua Bengio, and Geoffrey Hinton. "Deep learning." Nature 521.7553 (2015): 436-444.
12. Deng, Li, et al. "Recent advances in deep learning for speech research at Microsoft." Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on. IEEE, 2013.
13. Li, Chuanhao, et al. "Bearing fault diagnosis using fully-connected winner-take-all autoencoder." IEEE Access (2017).
14. Ince, Turker, et al. "Real-time motor fault detection by 1-D convolutional neural networks." IEEE Transactions on Industrial Electronics 63.11 (2016): 7067-7075.
15. Zhang, Wei, et al. "A deep convolutional neural network with new training methods for bearing fault diagnosis under noisy environment and different working load." Mechanical Systems and Signal Processing 100 (2018): 439-453.
16. Lee, Ki Bum, Sejune Cheon, and Chang Ouk Kim. "A Convolutional Neural Network for Fault Classification and Diagnosis in Semiconductor Manufacturing Processes." IEEE Transactions on Semiconductor Manufacturing 30.2 (2017): 135-142.
17. Goodfellow, Ian, Yoshua Bengio, and Aaron Courville. Deep learning. MIT press, 2016.
18. Glorot, Xavier, Antoine Bordes, and Yoshua Bengio. "Deep sparse rectifier neural networks." Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics. 2011.
19. Hochreiter, Sepp. "The vanishing gradient problem during learning recurrent neural nets and problem solutions." International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems 6.02 (1998): 107-116.